

# Parallel 3-dim fast Fourier transforms with load balancing of the plane waves

Xingyu Gao and Zeyao Mo

*Laboratory of Computational Physics,*

*Huayuan Road 6, Beijing 100088, P.R. China*

*Institute of Applied Physics and Computational Mathematics,*

*Fenghao East Road 2, Beijing 100094, P.R. China and*

*CAEP Software Center for High Performance Numerical Simulation,*

*Huayuan Road 6, Beijing 100088, P.R. China*

Jun Fang and Han Wang\*

*Institute of Applied Physics and Computational Mathematics,*

*Fenghao East Road 2, Beijing 100094, P.R. China and*

*CAEP Software Center for High Performance Numerical Simulation,*

*Huayuan Road 6, Beijing 100088, P.R. China*

## Abstract

The plane wave method is most widely used for solving the Kohn–Sham equations in first-principles materials science computations. In this procedure, the three-dimensional (3-dim) trial wave functions’ fast Fourier transform (FFT) is a regular operation and one of the most demanding algorithms in terms of the scalability on a parallel machine. We propose a new partitioning algorithm for the 3-dim FFT grid to accomplish the trade-off between the communication overhead and load balancing of the plane waves. It is shown by qualitative analysis and numerical results that our approach could scale the plane wave first-principles calculations up to more nodes.

**Keywords:** first-principles calculation, Kohn–Sham equation, plane wave, FFT, load balancing.

---

\*Electronic address: wang`han@iapcm.ac.cn

## I. INTRODUCTION

In the context of Density Functional Theory (DFT), solving the Kohn–Sham equation is the most time-consuming part of the first-principles materials science computations [1–3]. The plane wave method, which is a widely used numerical approach [4], could lead to a large-scale dense algebraic eigenvalue problem. This problem is usually solved by the iterative diagonalization methods such as Davidson’s[5], RMM-DIIS[3], LOBPCG[6], Chebyshev polynomial filtering subspace iteration[7], etc. The elementary operation of the iteration methods is the matrix-vector multiplication. Since the large-scale dense matrix is not suitable for explicit assembly, the matrix-vector multiplication is realized by applying the Hamiltonian operator on trial wave functions. The local term of the effective potential is one part of the Hamiltonian operator. In order to compute its action in a lower time complexity, we perform 3-dim FFT twice on one trial wave function in each matrix-vector multiplication.

There are three features to make the trial wave function’s FFT one of the most demanding algorithms to scale on a parallel machine. The first is the moderate sized FFT grid rather than a large one. The ratio of computation to communication of the parallel 3-dim FFT is of order  $\log N$  where  $N$ , the single dimension of the FFT grid, is usually  $\mathcal{O}(10^2)$  in most first-principles calculations of bulk materials. The second is the accumulated communication overhead led by many execution times corresponding to many wave functions in large-scale problems. Thousands of FFTs may run at each step of iterative diagonalization. The third is the all-to-all type communication required by the data transposes. This can limit the parallel scaling due to the large number of small messages in the network resulting in competition as well as latency issues.

It has already been recognized that making fewer and larger messages can speed up parallel trial wave functions’ FFTs. The hybrid OpenMP/MPI implementation [8, 9] can lead to fewer and larger messages compared to a pure MPI version. And a blocked version [9] performs a number of trial wave functions’ FFTs at the same time to aggregate the message sizes and reduce the latency problem.

In first-principles calculations, we should consider not only the parallel scaling of wave functions’ FFTs, but also the load balancing of intensive computations on the plane waves that expand the wave functions. The workload of these computations are usually inhomoge-

neously distributed on a standard 3-dim FFT grid. Thus a greedy algorithm is usually used to optimize the load balancing. However, this algorithm results in global all-to-all communications across all the processes, thus the latency overhead would grow in proportion to the number of processors and might contribute substantially to the total simulation time. Haynes et. al. [10] present a partitioning approach for the 3-dim FFT grid that minimizes the latency cost. Their method depends critically on the Danielson-Lanczos Lemma [11] and requires a particular data distribution, which limits the possibilities to improve the load balancing of the plane waves.

In this paper, we propose a new partitioning method for the 3-dim FFT grid, with which we need independent local all-to-all communications for each data transpose rather than one global all-to-all communication. With this communication pattern preserved, we develop the method to improve the load balancing by adjusting the data distribution among the working processors. By numerical examples, we show that although its load balancing is not as perfect as that of the greedy algorithm, the new approach can be more favorable for parallel scaling by making the fewer and larger messages. Hence we are allowed to accomplish the trade-off between the load balancing of the plane waves and communication overhead in the trial wave functions' FFTs. And such a trade-off could scale the plane wave first-principles calculations up to more nodes. With the proposed partitioning method, we design a compact parallel 3-dim FFT to reduce the amount of calculations and passing messages without loss of accuracy.

The rest of this paper is organized as follows. In Section 2 we introduce the elemental role of trial wave functions' FFTs in the plane wave method. In Section 3 we introduce the greedy algorithm for load balancing of the plane waves and analyze the resulting communication cost. In Section 4 we describe the new partitioning algorithms and implementations. In Section 5 we show the numerical results. The last section gives concluding remarks.

## II. ROLE OF TRIAL WAVE FUNCTIONS' FFT

In this section, we explain the elemental role of trial wave functions' FFTs in solving the Kohn-Sham equation using a plane wave basis set.

In the pseudopotential (norm-conserving [12] or ultrasoft [13] pseudopotential) setting

or the projector augmented wave (PAW) [14, 15] approach, the pseudo wave function  $\tilde{\Psi}_i$  satisfies the Kohn–Sham equation which looks like:

$$\left(-\frac{1}{2}\Delta + V_{\text{loc}} + V_{\text{nl}}\right) \tilde{\Psi}_i = \epsilon_i S \tilde{\Psi}_i, \quad (1)$$

where  $-\frac{1}{2}\Delta$  is the kinetic energy operator,  $V_{\text{loc}}$  the local potential,  $V_{\text{nl}}$  the nonlocal term, and  $S$  the overlapping operator. In the case of the norm-conserving pseudopotential,  $S$  could simply be interpreted as the identity operator. In this paper, we refer to the pseudo wave function simply as the wave function.

We use always the periodic boundary condition and expand the wave functions in plane waves:

$$\tilde{\Psi}_{n\mathbf{k}}(\mathbf{r}) = \sum_{\mathbf{G}} \tilde{\Psi}_{n\mathbf{k}}(\mathbf{G}) e^{-i(\mathbf{k}+\mathbf{G})\cdot\mathbf{r}}, \quad (2)$$

where the  $\mathbf{k}$ 's are vectors sampling the first Brillouin zone,  $n$  is an index of the energy level with given  $k$ , and  $\mathbf{G}$ 's are the reciprocal lattice vectors. The expansion (2) only includes the plane waves satisfying

$$|\mathbf{k} + \mathbf{G}| < \sqrt{2E_{\text{cut}}} \equiv G_{\text{cut}}. \quad (3)$$

In the plane wave discretization of one large-scale problem, the Hamiltonian matrix should never be assembled explicitly. Instead, iterative diagonalization techniques are employed together with the implicit matrix-vector multiplication that is realized as the action of the Hamiltonian operator on the trial wave functions. It is noticed that the local potential is diagonal in the real space. In order to obtain efficiently the action of the local potential on the wave function, we should first transform  $\tilde{\Psi}_{n\mathbf{k}}(\mathbf{G})$  to the real space representation  $\tilde{\Psi}_{n\mathbf{k}}(\mathbf{r})$  by one FFT, multiply with the local potential term, and then transform the product back to the reciprocal space. Consequently, two 3-dim FFTs are required by each action on a trial wave function.

### III. THE LOAD BALANCING ISSUE AND THE GREEDY ALGORITHM

#### A. The load balancing issue

As mentioned in the previous section, the plane waves are truncated at a certain cut-off radius  $G_{\text{cut}}$ . Since the charge density  $\rho$  is the sum of squares of the wave functions in the

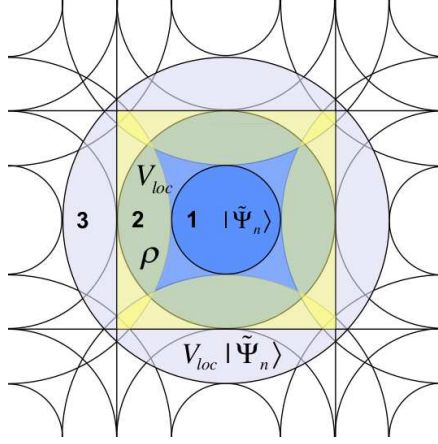


FIG. 1: A two dimensional sketch of the wrap-around errors in the periodic reciprocal space. The wave functions  $|\Psi\rangle$  is sampled within a sphere with the radius  $G_{\text{cut}}$  (the innermost circle 1). The charge density  $\rho$  and the local potential  $V_{\text{loc}}$  are defined inside a sphere with the radius  $2G_{\text{cut}}$  (circle 2). We would require a sphere with the radius  $3G_{\text{cut}}$  to accurately estimate the operation of the local potential on the wave function. If we apply a smaller FFT grid with only  $2G_{\text{cut}}$ , the artificial wrap-around error between  $2G_{\text{cut}}$  and  $3G_{\text{cut}}$  would occur and be folded back into the 2nd circle due to the periodicity. Hence it is sufficient to approximate the wave functions and gradients correctly in circle 1.

real space, the corresponding cut-off radius for the charge density is  $2G_{\text{cut}}$ . The cut-off radius of the local potential  $V_{\text{loc}}$  can be regarded the same as that of  $\rho$ , because  $V_{\text{loc}}$  is a functional of  $\rho$ . Thus, the cut-off radius of  $V_{\text{loc}}\tilde{\Psi}_{n\mathbf{k}}$  is  $3G_{\text{cut}}$ . It should be noted that the Kohn–Sham equation (1) is discretized by the plane waves lying in the sphere of radius  $G_{\text{cut}}$ . As illustrated in Fig. 1, it is sufficient to take the FFT grid with only  $2G_{\text{cut}}$  for preventing the wave functions from the wrap-around error and solving (1) correctly.

On one hand, we compute the operation of the local potential on the trial wave functions by the 3-dim FFTs on the standard grid determined by the cut-off radius  $2G_{\text{cut}}$ . On the other hand, we carry out intensive computations time complexities of which are in proportion to the number of the plane waves in a sphere of radius  $G_{\text{cut}}$ , including the assembly of the matrix on the subspace, the orthogonalization of wave functions, and the actions of other parts of the Hamiltonian operator. Thus the workload of the intensive calculations is not homogeneously distributed on the grid. In partitioning the grid, one should consider not only the parallel scaling of 3-dim FFTs, but also the load balancing issue of the plane waves

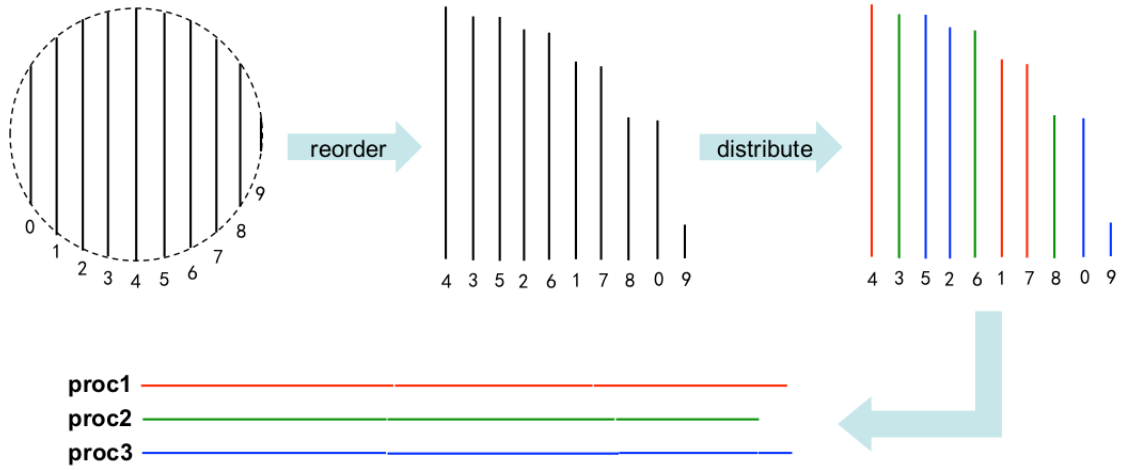


FIG. 2: A two dimensional illustration of the greedy algorithm for building the reciprocal space layout.

calculations.

### B. The greedy algorithm

One 3-dim FFT consists of three successive sets of 1-dim FFTs along the  $x$ ,  $y$  and  $z$  directions. For each set of 1-dim FFT, the data layout guarantees that each processor holds the complete columns of data along the FFT direction. Therefore, there are three data layouts of the 1-dim FFTs along the  $x$ ,  $y$  and  $z$  directions. We call them the reciprocal space, intermediate and real space layouts, respectively.

The greedy algorithm is used to build the reciprocal space layout for the sake of load balancing. In the reciprocal space layout, each processor holds the complete columns along the  $x$  direction. The workload of each complete column is estimated by the number of plane waves within the cut-off radius  $G_{\text{cut}}$ . As illustrated by Fig. 2, we sort these columns in the descending order of workload and distribute the individual columns among processors in a round robin fashion. In this way, the reciprocal space layout is established and each processor holds a set of complete columns with approximately equal workload. Due to the uniform distribution of workload, we could directly distribute the individual columns in a cyclic way to establish the intermediate and real space layouts.

### C. The communication pattern and overhead

When finishing one set of 1-dim FFTs along one direction, we transpose the data from the current layout to the next one for the successive set of 1-dim FFTs. The first transpose is between the reciprocal space layout and intermediate layout, while the second one is between the intermediate layout and real space layout. With the reciprocal space layout established by the greedy algorithm, the first transpose typically requires the all-to-all communication. The second transpose may require no communications if each processor holds complete planes (perpendicular to the  $x$  direction), or limited local communications if each processor has a section of a plane.

In general, the overhead of the all-to-all data communication mainly consists of two parts: the data transmission and the network latency. The transmission cost is proportional to the total size of the data packets, and inversely proportional to the internode bandwidth denoted by  $\beta$ . The latency cost is proportional to the number of data transmissions initiated. We denote the latency of one data transmission by  $\alpha$ . It worth noting that  $\alpha$  and  $\beta$  are defined for the situation that a node is sending a data packet to another node and simultaneously receiving a packet from another node.

Without loss of generality, we assume that the all-to-all communications is implemented by the pairwise data exchanges. Alternative implementations can be found in Ref. [16]. Thus the all-to-all communication of  $p$  processors can be achieved in  $p - 1$  phases. In each phase, each processor simultaneously sends a data packet to one processor and receives a packet from another (usually different) processor. Though the sizes of data packets are not uniform (in an Alltoallv operation), the average size of one packet can be estimated by  $\mu N_{\text{FFT}}/p^2$ , where  $\mu$  is the size of a single element (typically 16 bytes for a double precision complex data type), and  $N_{\text{FFT}}$  is number of the FFT grids. Hence we estimate the total cost of one all-to-all communication as:

$$t_1 = (p - 1) \left( \alpha + \frac{\mu N_{\text{FFT}}}{\beta p^2} \right). \quad (4)$$

For a fixed  $N_{\text{FFT}}$ , the latency overhead grows linearly with respect to the number of processors, which will probably result in a limited parallel scaling.

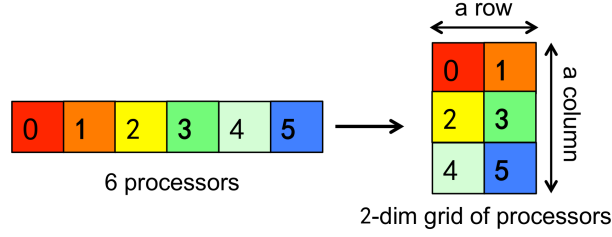


FIG. 3: The illustration of a  $3 \times 2$  grid of processors.

#### IV. THE NEW PARTITIONING ALGORITHM AND ITS IMPLEMENTATION

In this section, we present a new partitioning algorithm of the 3-dim FFT grid to avoid global all-to-all communications required by the data transposes, so that the latency cost is alleviated at a cost of small loss of load balancing.

##### A. The idea of the basic algorithm

We assume that the number of processors  $p$  can be factorized by  $m \times n$ , where the difference between  $m$  and  $n$ , i.e.  $|m - n|$ , should be as small as possible. Then the  $p$  processors are grouped into  $m$  rows by  $n$  columns (a  $3 \times 2$  case is illustrated by Fig. 3). Take the reciprocal space layout for example. We distribute the complete columns of data along the  $x$  direction following two rules: 1, The data columns with the same  $y$ -index are distributed within the same column group of processors. 2, The data columns with the same  $z$ -index are distributed within the same row group of processors. The intermediate and the real space data layout can be established in a similar way. The only restrictions are that the intermediate layout shares the same data distribution with the reciprocal space layout along the  $z$  direction and with the real space layout along the  $x$  direction. In another word, each  $xy$  plane in the intermediate layout is distributed among the same row of processors as the  $xy$  plane in the reciprocal space layout with the same  $z$  index, and each  $yz$  plane in the intermediate layout is distributed among the same column of processors as the  $yz$  plane in the real space layout with the same  $x$  index.

A direct implementation of the algorithm is to distribute the data columns in a cyclic fashion. As an illustration, we show how to use it to distribute a 3-dim FFT grid of size  $5 \times 5 \times 5$  among 6 processors. The 6 processors are grouped into 3 rows by 2 columns, as



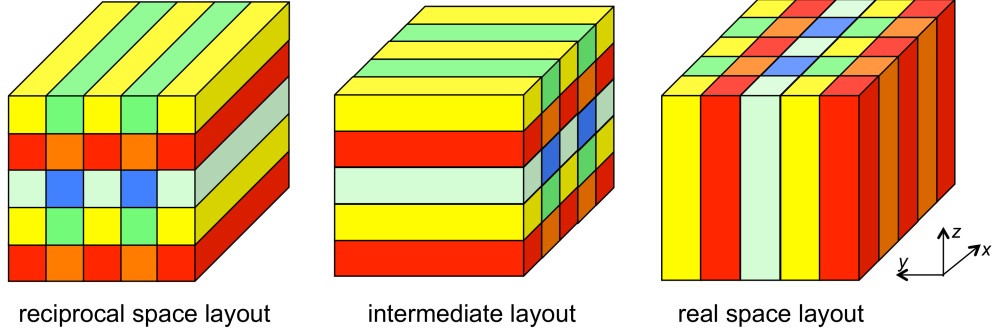


FIG. 4: The resulting three layouts by using the basic algorithm to partition the 3-dim FFT grid of  $5 \times 5 \times 5$  among 6 processors.

shown by Fig. 3. The reciprocal space, intermediate and the real space layouts established by our method are shown, from left to right respectively, by Fig. 4. We call this implementation the “basic algorithm” in this manuscript.

In general, the first data transpose (between reciprocal space and intermediate layouts) requires  $m$  local all-to-all communications which can be carried out *independently* within row groups of  $n$  processors. Similarly, the second transpose (between intermediate and real space layouts) requires  $n$  local all-to-all communications which can be carried out *independently* within column groups of  $m$  processors. As shown by Fig. 4, the first data transpose requires local all-to-all communications within row groups of two processors, and the second data transpose requires local all-to-all communications within column groups of three processors. When  $m$  and  $n$  are roughly equal to  $\sqrt{p}$ , the communication overhead can be estimated as:

$$t_2 = (\sqrt{p} - 1) \left( \alpha + \frac{\mu N_{\text{FFT}}}{\beta p \sqrt{p}} \right). \quad (5)$$

Compared with the estimated cost (4) of the global all-to-all communication, the growth rate of the latency cost with respect to the number of processors is decreased from  $p$  to  $\sqrt{p}$ .

It should be clarified that both (4) and (5) are used to qualitatively illustrate how the communication overhead is decreased rather than to give an quantitative interpretation of actual running time. Compared with a global all-to-all communication, the new local all-to-all communications make fewer ( $p(\sqrt{p} - 1)$  v.s.  $p(p - 1)$ ) and larger ( $\mu N_{\text{FFT}}/(p\sqrt{p})$  v.s.  $\mu N_{\text{FFT}}/p^2$ ) messages, which alleviates the competition as well as latency issues in the network. So the proposed partitioning algorithm offers the prospect of scaling the plane wave first-principles calculations up to more nodes.

## B. The improved algorithm considering the load balancing

The local all-to-all communications can be kept if the aforementioned partitioning restrictions are satisfied, i.e. the intermediate layout shares the same data distribution with the reciprocal space layout along the  $z$  direction, and the same data distribution with the real space layout along the  $x$  direction. So we are allowed to improve the reciprocal space layout considering the load balancing issue.

Here we present one possibility: Firstly the workload of each  $xz$ -plane and  $xy$ -plane is estimated by the number of plane waves in the sphere of radius  $G_{\text{cut}}$ . Secondly, the  $xz$ -planes are sorted in the descending order with respect to the workload, and then the reordered  $xz$ -planes are distributed to the column groups of processors in a round robin fashion. Finally, the  $xy$ -planes are sorted in the descending order with respect to the workload, and then the reordered  $xy$ -planes are distributed to the row groups of processors in a round robin fashion. We will call this implementation the “improved algorithm” in this manuscript.

## C. The compact 3-dim FFT

As we have discussed in Sec. III A, the standard 3-dim FFT grid is determined by the cut-off radius  $2G_{\text{cut}}$ , while the wave functions are represented by plane waves within the sphere of radius  $G_{\text{cut}}$  (Fig. 5 (a)). Thus, one can pick up the complete  $x$ -data columns that intersect with the sphere to perform 1-dim FFTs along the  $x$  direction, because all other  $x$ -data columns contain only vanishing values. All the selected columns constitute a cylinder, as shown in Fig. 5 (b). After the  $x$  direction FFTs, only this cylinder contains non-zero data. Then one can select the complete  $y$ -data columns that intersect with the cylinder to perform the  $y$  direction 1-dim FFTs, and the resulting non-zero-data region is a cuboid, as shown in Fig. 5 (c). The last set of 1-dim FFTs along the  $z$  direction is performed on the whole cube shown as in Fig. 5 (d). Such a compact 3-dim FFT can also reduce the amount of passing messages and calculations compared to the standard 3-dim FFT implementation that performs 1-dim FFTs on all  $x$  and  $y$  data columns in the cube.

If only the  $\Gamma$ -point is used for the  $\mathbf{k}$ -point sampling, we can implement a real mode where the reciprocal space and intermediate layouts can be cut by half since we take into account that  $\tilde{\Psi}_n(\mathbf{G}) = \tilde{\Psi}_n^*(-\mathbf{G})$ .

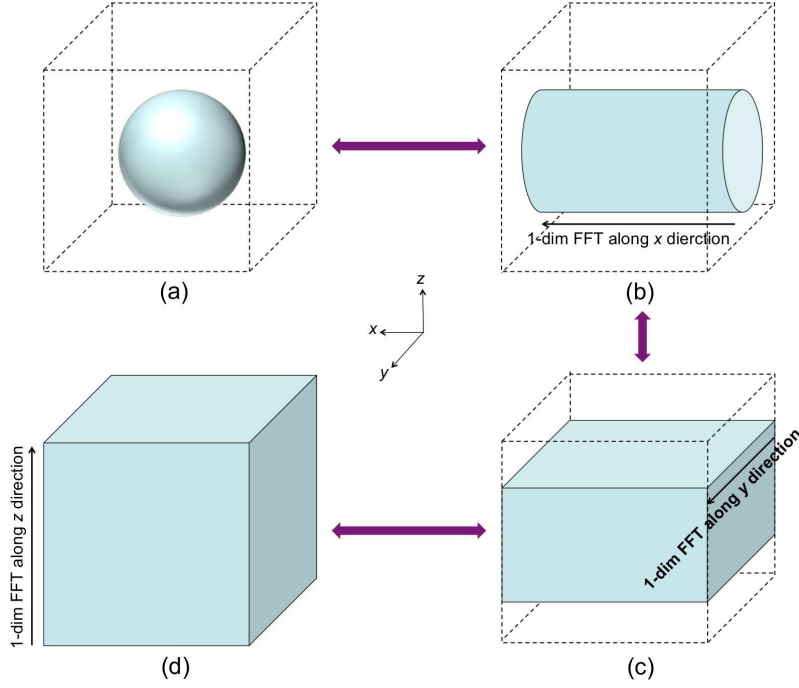


FIG. 5: The illustration of a compact 3-dim FFT. (a): The aqua region presents the sphere of cut-off radius  $G_{\text{cut}}$ . (b) – (d): The aqua regions present union of all data columns selected to perform the  $x$ ,  $y$  and  $z$  1-dim FFTs, respectively.

## V. NUMERICAL RESULTS

We implement the parallel compact 3-dim FFT for the trial wave functions in the in-house plane wave code package CESSP developed on the infrastructure JASMIN [17]. Our implementation is a pure MPI version including the greedy algorithm, basic algorithm and improved algorithm. With these partitioning algorithms, the parallel scaling of solving the Kohn–Sham equation (1) in PAW approach are tested on a domestic parallel machine. Each node of the machine consists of 2 Intel Xeon E5540 CPUs (8 cores) and the nodes are connected by the infiniband with double data rate (DDR).

The testing system, which is sampled by only the  $\Gamma$ -point, is defined on an FCC (face centered cubic) supercell consisting of 500 Al (aluminum) atoms. The self-consistent field iteration runs 7 cycles, and in each cycle the RMM-DIIS algorithm [15] is employed to solve the lowest 1001 eigenstates. In this process, 38038 FFTs of the trial wave functions are executed one by one. The size of the 3-dim FFT grid is  $80 \times 80 \times 80$ , while the sphere of radius  $G_{\text{cut}}$  consists of 35160 plane waves for the expansion of wave functions.

TABLE I: Comparison on the parallel scaling of three partitioning algorithms.

Greedy algorithm					Our basic algorithm				Our improved version			
# MPI proc.	# data transposes	comm. time of trial wfs' FFTs (s)	total wall time (s)	parallel scaling efficiency	# data transposes	comm. time of trial wfs' FFTs (s)	total wall time (s)	parallel scaling efficiency	# data transposes	comm. time of trial wfs' FFTs (s)	total wall time (s)	parallel scaling efficiency
16	38038	7.61	522.27	1.00	76076	15.59	551.77	1.00	76076	16.61	551.07	1.00
24		9.28	373.82	0.93		13.40	388.44	0.95		14.42	370.30	0.99
36		35.11	291.53	0.80		10.36	275.51	0.89		9.78	256.14	0.96
64		117.32	303.31	0.43		8.62	183.21	0.75		7.79	176.13	0.78

TABLE II: Comparison on the load balancing of three partitioning algorithms

The greedy algorithm			Our basic algorithm		Our improved version	
# MPI proc.	max. number of plane waves on single proc.	mim. number of plane waves on single proc.	max. number of plane waves on single proc.	mim. number of plane waves on single proc.	max. number of plane waves on single proc.	mim. number of plane waves on single proc.
16	2209	2190	2327	2068	2261	2130
24	1469	1461	1647	1294	1549	1398
36	982	972	1210	814	1016	952
64	554	544	641	454	643	492

In all tests, we launch 8 pure MPI processes per node and count the number of data transposes, the total wall time as well as communication time of the trial wave functions' FFTs. The results of the tests are summarized in Tab. I. In the greedy algorithm, no data transposes are required between the intermediate and real space layouts since the intermediate layout holds the complete  $yz$ -planes on each processor. So the greedy algorithm needs half number of data transposes as our new algorithms. Nevertheless, as shown by Tab. I, with increasing number of processors, the greedy algorithm leads to a rapid growth in the communication cost, which finally takes more than one third of the total computational cost, while the new algorithms (both the basic and its improved versions) can effectively suppress the growth in the communication cost. When the number of processors is less than 24, the greedy algorithm is preferable, while when the number of processors is more than 24, our algorithms could provide better overall performance.

In Tab. II we represent the load balancing in the simulations by comparing the maximum and minimum numbers of plane waves distributed on a single processor. It is obvious that the load balancing of the greedy algorithm is almost perfect. Although not as perfect as the greedy algorithm, the load balancing of our improved algorithm is still acceptable. Comparing with the basic algorithm, the improved version can effectively reduce the gap

between the maximum and minimum number of plane waves on a processor. Combining Tab. I and II together, we present an example of the trade-off between the load balancing and communication cost: the greedy algorithm has best load balancing but could lead to very limited parallel scaling; our new algorithms would achieve much better scaling at a moderate loss of load balancing.

## VI. CONCLUSION

We present a new partitioning algorithm for the 3-dim FFT grid used in the plane wave first-principles calculations. Compared with the greedy algorithm biased toward load balancing of the plane wave computations, our approach primarily suppresses the growth in communication overhead with respect to increasing number of processors by performing local all-to-all communications during data transposes. Then we adjust the data distribution to improve the load balancing with the communication pattern preserved. In the numerical examples, we present a trade-off: a much lower communication overhead on a relatively large number of processors is achieved at a moderate loss of load balancing. By using the new algorithm, we could scale the whole plane wave codes up to more processors than the greedy algorithm. For a better performance, our approach can be seamlessly combined with other techniques such as the hybrid OpenMP/MPI implementation or simultaneously performing a large number of FFTs.

## Acknowledgment

The authors would like to thank Hong Guo and Zhang Yang for useful discussions. And the first author is especially thankful to Xiaowen Xu for his encouragement. This work was partially supported by the National Science Foundation of China under Grants 91430218 and 61300012, and the National High Technology Research and Development Program of China under Grant 2015AA01A304. H.W. is supported by the National Science Foundation of China under Grants 11501039 and 91530322.

---

[1] W. Kohn and L. J. Sham, Phys. Rev. **140**, A1133 (1965).

- [2] G. Kresse and J. Furthmüller, Comput. Mat. Sci. **6**, 15 (1996).
- [3] G. Kresse and J. Furthmüller, Phys. Rev. B **54**, 11169 (1996).
- [4] M. C. Payne, M. P. Teter, and D. C. Allan, Rev. Mod. Phys. **64**, 1045 (1992).
- [5] B. Liu, in *Numerical Algorithms in Chemistry: Algebraic Methods*, edited by E. Moler and I. Shavitt (Lawrence Berkley Lab. Univ. of California, 1978), p. 49.
- [6] A. V. Knyazev, SIAM J. Sci. Comput. **23**, 517 (2001).
- [7] Y. Zhou, Y. Saad, M. L. Tiago, and J. R. Chelikowsky, J. Comput. Phys. **219**, 172 (2001).
- [8] S. Goedecker, M. Boulet, and T. Deutsch, Comput. Phys. Comm. **154**, 105 (2003).
- [9] A. Canning, J. Shalf, N. J. Wright, S. Anderson, and M. Gajbe, in *The 9th International Conference on Scientific Computing* (2012).
- [10] P. D. Haynes and M. Côté, Comput. Phys. Comm. **130**, 130 (2000).
- [11] G. C. Danielson and C. Lanczos, J. Franklin Inst. **233**, 365 (1942).
- [12] D. R. Hamann, M. Schlüter, and C. Chiang, Phys. Rev. Lett. **43**, 1494 (1979).
- [13] D. Vanderbilt, Phys. Rev. B **41**, 7892 (1990).
- [14] P. E. Blöchl, Phys. Rev. B **50**, 17953 (1994).
- [15] G. Kresse and J. Joubert, Phys. Rev. B **59** (1999).
- [16] L. Rao, Y. Zhang, and Y. Li, Computer Science **37**, 186 (2010).
- [17] Z. Mo, A. Zhang, X. Cao, Q. Liu, X. Xu, H. An, W. Pei, and S. Zhu, Frontiers of Computer Science in China **4**, 480 (2010).